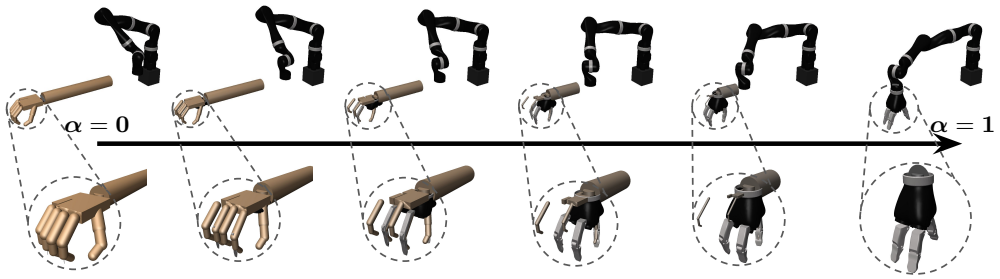

Figure 1: **Continuous hand-to-robot evolution for Jaco robot.** We show one evolution path from human dexterous hand ($\alpha = 0$) to a commercial Jaco robot with three-finger Jaco gripper ($\alpha = 1$). From left to right, we show intermediate robots along the path. Zoom in for better view.

HERD: Continuous Human-to-Robot Evolution for Learning from Human Demonstration – Supplementary Material

Xingyu Liu **Deepak Pathak** **Kris M. Kitani**
Carnegie Mellon University

A Overview

In this document, we provide additional information as presented in the main paper. We present additional technical details of our solution to human-to-robot continuous evolution in Section B. We introduce our proposed method for learning expert dexterous robot policy from human visual demonstration in Section C. We provide details on the hyperparameter settings of our experiments in Section D. Lastly, in Section E, we discuss further on related works and problems.

B Details on Hand-to-Robot Continuous Evolution

Virtual Confinement of the Robot End Effector The implementation of the dexterous hand robot simulation usually assumes the elbow is able to move freely in all 6 degrees. This is modeled as the elbow being connected by six virtual joints that are attached to a fixed mount point in space. However, in our solution of human-to-robot evolution, the elbow of the dexterous robot is modeled as being connected to the robot end effector. If the robot end effector is also allowed to move freely at the start of the evolution, a well-trained policy on the original dexterous robot will be ruined by the motion of the robot end effector, because the original policy assumes that the virtual joints are mounted to a fixed base. The policy transfer will fail even before the training on the continuously evolving robots starts.

We propose a mechanism named *virtual confinement* to address this. Virtual confinement exerts restrictions on the motion range of the robot end effector. At the start of the evolution, the allowed motion range of the robot end effector in all degrees of freedom is set to 0 which means the robot end effector is completely frozen. This means the virtual free joints of the dexterous hand robot are equivalently still mounted on a fixed base so that the original dexterous hand robot expert policy can be seamlessly imported to the combined robot. The motion range of the robot end effector gradually increases with the evolution progress and eventually allows the robot to move freely in the space of the environment. The virtual confinement does not correspond to any entity in the simulation environment but is modeled as several virtual joints mounted in a virtual base in space. The position of the virtual base is the same as the initial position of the robot end effector. The restriction on the motion range of the virtual joints can be implemented by the `equality` mechanism in MuJoCo

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

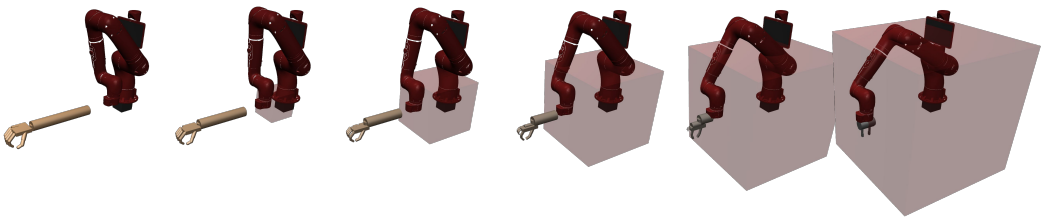

Figure 2: **Continuous evolution of the virtual confinement of the robot end effector.** The red semitransparent 3D box denotes the 3D space that the robot end effector is allowed to move to. Initially the end effector is completely frozen in its motion. The allowed 6D motion range gradually increases with evolution.

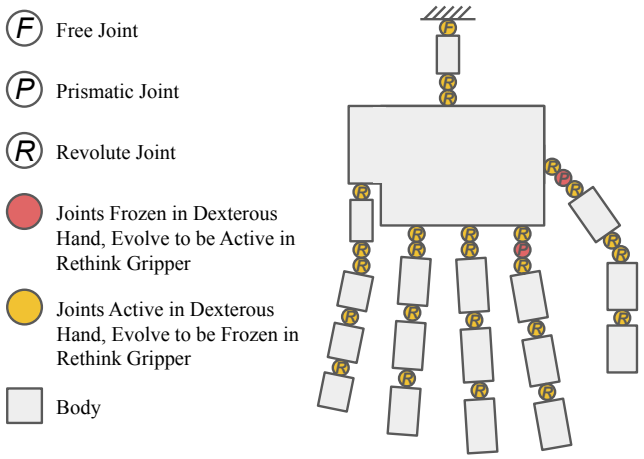

Figure 3: **Kinematic tree of dexterous hand robot.** All revolute and free joints will gradually freeze during evolution. The two prismatic joints are initially frozen and evolve to be active.

Engine [1]. We use the proposed virtual confinement mechanism in all experiments in the main paper. The virtual confinement mechanism is illustrated in Figure 2.

Gripper Finger Synchronization The fingers of commercial robot grippers are usually synchronized in motion as position servo so that the grasping action is controlled by a single signal. On the other hand, the motion of the human hand fingers are asynchronous and independent from each other. To interpolate such behavior, we add additional position servo joints to the human hand. During robot evolution from human hand to the target robot, the motion range of the position servos gradually increase while the range of the independent joints gradually shrink to zero.

Disabling Collision Since the Saywer robot end effector is not originally designed to be mounted with a dexterous hand robot, it is likely that the dexterous hand robot and the target Sawyer robot can collide during simulation unexpectedly and cause unstable behavior. Therefore, during simulation, we manually disable all pairs of collisions between the human hand bodies and robot bodies. Disabling collisions can be implemented by the `exclude` mechanism in MuJoCo Engine [1].

Human-to-robot Evolution Specifics We illustrate the kinematic tree of the dexterous hand robot in Figure 3. For Sawyer robot with Rethink gripper, during evolution, all revolute joints and the elbow free joint gradually freeze to have a range of 0. On the other hand, the two synchronized prismatic joints are initially frozen with a range of 0, and their range gradually increases until the same full range as the Rethink gripper. During the evolution, all bodies of the middle finger, ring finger and little finger gradually shrink to zero-size and disappear. The shapes of the bodies of the first finger, thumb and palm change continuously and become the same shape as the Rethink gripper eventually. The robot parameters of the Sawyer robot where the dexterous hand robot is attached to are not changed, except the range of virtual confinement which has the same evolution progress as the `elbow_joint_range`.

ID	Robot Parameter	ID	Robot Parameter	ID	Robot Parameter
1	th_proximal_length	15	lf_distal_length	29	mf_shape
2	th_middle_length	16	th_joint_4_range	30	rf_shape
3	th_distal_length	17	th_joint_3_range	31	lf_shape
4	ff_proximal_length	18	th_joint_2_range	32	th_knuckle_position
5	ff_middle_length	19	th_joint_1_range	33	ff_knuckle_position
6	ff_distal_length	20	th_joint_0_range	34	mf_knuckle_position
7	mf_proximal_length	21	ff_joint_3_range	35	rf_knuckle_position
8	mf_middle_length	22	ff_joint_2_range	36	lf_knuckle_position
9	mf_distal_length	23	ff_joint_1_range	37	palm_shape
10	rf_proximal_length	24	ff_joint_0_range	38	wrist_joints
11	rf_middle_length	25	th_slide_joint	39	arm_length
12	rf_distal_length	26	ff_slide_joint	40	elbow_joint_range
13	lf_proximal_length	27	th_shape		
14	lf_middle_length	28	ff_shape		

Table 1: **Space of robot parameter evolution for two-finger Sawyer robot as target.** th, ff, mf, rt and lf represent thumb, first finger, middle finger, ring finger and little finger respectively. The robot evolution is described by $D = 40$ independent parameters in evolution in total.

ID	Robot Parameter	ID	Robot Parameter	ID	Robot Parameter
1	th_proximal_length	15	lf_distal_length	29	th_shape
2	th_middle_length	16	th_joint_4_range	30	ff_shape
3	th_distal_length	17	th_joint_3_range	31	mf_shape
4	ff_proximal_length	18	th_joint_2_range	32	rf_shape
5	ff_middle_length	19	th_joint_1_range	33	lf_shape
6	ff_distal_length	20	th_joint_0_range	34	th_knuckle_position
7	mf_proximal_length	21	ff_joint_3_range	35	ff_knuckle_position
8	mf_middle_length	22	ff_joint_2_range	36	mf_knuckle_position
9	mf_distal_length	23	ff_joint_1_range	37	rf_knuckle_position
10	rf_proximal_length	24	ff_joint_0_range	38	lf_knuckle_position
11	rf_middle_length	25	rf_joint_3_range	39	palm_shape
12	rf_distal_length	26	rf_joint_2_range	40	wrist_joints
13	lf_proximal_length	27	rf_joint_1_range	41	arm_length
14	lf_middle_length	28	rf_joint_0_range	42	elbow_joint_range

Table 2: **Space of robot parameter evolution for three-finger Jaco robot as target.** th, ff, mf, rt and lf represent thumb, first finger, middle finger, ring finger and little finger respectively. The robot evolution is described by $D = 42$ independent parameters in evolution in total.

The evolution process for the three-finger Jaco robot is slightly different from Sawyer. In Jaco robot, the synchronized prismatic joints are the three revolute joints of the gripper. In order to model the interpolation of the synchronized joints, similar to Sawyer robot solution, we add three additional synchronized position servo joints. The three synchronized revolute joints are initially frozen with a range of 0, and their range gradually increases until the same full range of the Jaco gripper. The evolution process for Jaco robot is illustrated in Figure 1.

As mentioned in the main paper, our evolution solution includes the changing of D independent robot parameters where $D = 40$ for Sawyer robot and $D = 42$ for Jaco robot. We illustrate the robot parameters in Tables 1 and 2. The changing robot parameters include body shapes and mass, and joint ranges and damping etc. The ranges of all changes in robot parameters are linearly normalized to $[0, 1]$ so that we can map the evolution progress of robot parameters to $[0, 1]^D$.

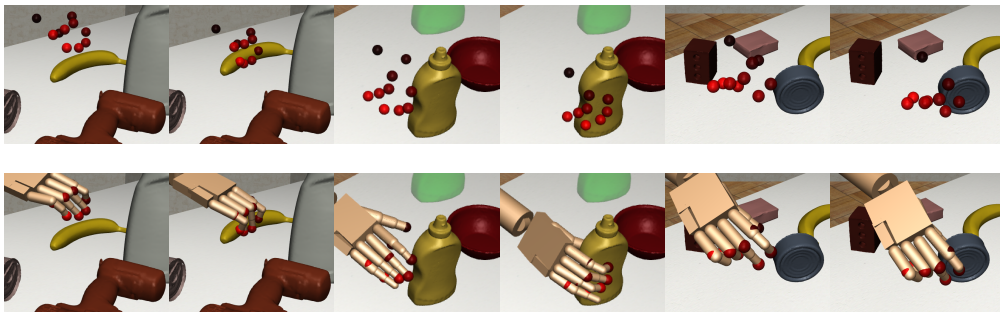

Figure 4: **Visualization of inverse kinematics (IK).** We use red spheres to show the hand 3D joint positions estimated by the vision model [4]. The second row is the result of using IK to fit the dexterous robot keypoints (i.e. finger tips and knuckles) to the corresponding human hand finger joint keypoints.

C Learning Expert Dexterous Hand Robot Policy from Visual Human Demonstrations

Obtaining an expert policy on dexterous hand robot from human demonstration is the first step of our proposed pipeline. In Section 3.1 of the main paper, we provide a brief description of our approach. In this section, we provide more details on the technical approach of learning expert dexterous hand robot policy from *visual* human demonstration. It includes the reconstruction of simulation states from visual inputs using vision models, and learning through generated reverse curriculum from human demonstrations. We use DexYCB dataset [2] as an example to describe our method.

C.1 Simulation States from Visual Demonstration

Given the visual scene of human manipulation demonstration, the goal of this step is to reconstruct the simulation environment by recovering the underlying states from visual inputs so that the dexterous human robot can learn the same behavior as demonstrated by the human hand. The states include the 6D poses of the objects and the joint poses of the human hand.

Object 6D Pose We use off-the-shelf models of CozyPose [3] to estimate the object 6D poses. CozyPose is a multi-view method for 6D object pose estimation. The object simulation states can be easily reconstructed by placing the scanned mesh model in the estimated 6D pose.

Dexterous Robot Pose We use off-the-shelf models of HRNet32 [4] to estimate the human hand joint poses. Due to the difference in shapes between the human hand and the dexterous robot, if we directly set the pose of the dexterous robot to be the estimated human hand pose, there could be an unwanted mismatch in the object grasping including penetration of object models, as pointed out by [5]. Therefore, we use inverse kinematics (IK) to minimize the average 3D distance between the positions of several dexterous robot joint keypoints and the corresponding human hand joint keypoints. We use ten joint keypoints, namely the tips and knuckles of the five fingers. To prevent the dexterous robot from penetrating the object mesh, we compute the volume of the intersection between the robot body mesh and the object mesh, and add the negative volume to the optimization objective of IK. We use the Powell algorithm [6] as the optimization algorithm for IK. We visualize several results of IK in Figure 4.

C.2 Curriculum for Learning Expert Policy on Dexterous Robot

Given the reconstructed simulation environment with the state-only trajectory estimated from vision models, our next goal is to train the expert policy on the dexterous hand robot to re-produce the same or similar behaviors as demonstrated by the human from visual input. Though recent literature [5, 7] have reported success on this problem, their implementations have not been released at the time of writing. We propose our solution to this task which is based on curriculum learning and reinforcement learning.

Our philosophy is to design a series of curricula to decompose the problem into multiple easier problems. Since we have access to the full state trajectory in DexYCB dataset demonstrations [2],

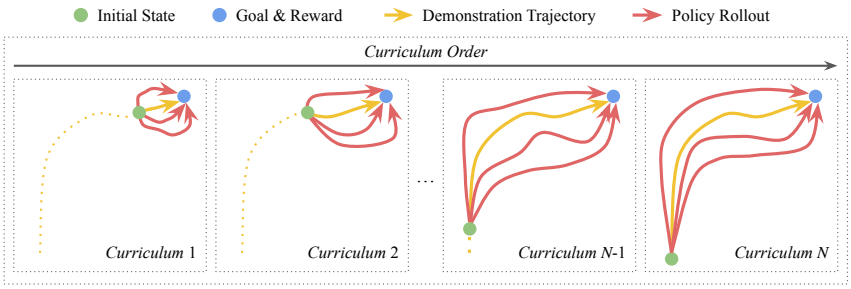

Figure 5: Curriculum learning of expert manipulation policy on dexterous hand robot.

we are able to reset the initial states of the simulation environment to the states at any timestamp in the demonstration trajectory.

During the first curriculum, the initial states of each simulation epoch are reset to be along the demonstrated trajectory while being very close to the set of goal states. With exploration, the agent can easily find trajectories of state-action pairs that lead to the goal. A policy can be trained with the explored trajectories. Then in the next curriculum, the initial states of each simulation epoch are still reset to be along the demonstration trajectory but move by a small distance in the reverse direction of demonstration compared to the first curriculum. During exploration, as long as the dexterous robot can enter the state regions explored and trained in the previous curriculum, it is easy for the trained policy to produce a state-action trajectory leading to the goal states. Then the policy is updated by training on the explored trajectories in the new curriculum. The above iteration of "initial state reverse movement + policy update" is repeated multiple times until reaching the desired initial state of the task. In this way, we will be able to decompose the difficult long-horizon continuous control tasks such as object manipulation into multiple easy tasks and finally solve the problem. The above learning process is illustrated in Figure 5.

Since we only use human demonstrations to reset epoch initial states, our curriculum strategy is able to deal with the task of one-shot learning from demonstration such as the case in the DexYCB dataset [2] where there is only one human demonstration video in each scene. We are also able to deal with noisy demonstrations such as the case of DexYCB dataset where the object and human states are estimated by vision models and could be noisy.

We point out that the similar idea of resetting simulation states for reverse curriculum generation can be found in previous work [8]. Different from [8], our curricula are generated along the human demonstration trajectory. Therefore, our curriculum design is more controllable and can train policies with behavior much closer to the human demonstration.

D Hyperparameter and Training Details

In this section, we present the hyperparameters and training procedures of our robot evolution and policy optimization. We use PyTorch [9] as our deep learning framework and NPG [10] as the RL algorithm in all experiments. To fairly compare against REvolveR [11], we use the same evolution progression step size ξ as [11] in the experiments. Since we use sparse reward setting in all experiments, in practice we use threshold in success rate q instead of episode reward in line 5 of Algorithm 1 of the main paper. The hyperparameters are illustrated in Table 3.

E More Discussions

Evolution Path being Optimal? We point out that the proposed DEPS algorithm for joint optimization of robot evolution path and policy does not guarantee to find the optimal robot evolution path. In fact, finding such an optimal path is an NP-complete problem, similar to the ordinary path planning problem. However, we argue that our algorithm does provide a working solution that allows us to find an optimized and efficient robot evolution path for transferring the policy than naïve linear evolution path, as shown by the reported experiment results.

Relation to Neural Architecture Search Our problem is closely related to the problem of Neural Architecture Search (NAS) [12, 13, 14] where we can view the intermediate robot in our problem as

Hyperparameter	Value
RL Discount Factor γ	0.995
GAE	0.97
NPG Step Size	0.0001
Policy Network Hidden Layer Sizes	(32,32)
Value Network Hidden Layer Sizes	(32,32)
Simulation Epoch Trajectory Length	200
RL Traning Batch Size	12
Evolution Progression Step Size ξ	0.03
Number of Sampled $\boldsymbol{\delta}_i$ for Jacobian Estimation n	72
Evolution Direction Weight Factor λ	1.0
Sample Range Shrink Ratio λ_1	0.995
Success Rate Threshold q	0.667

Table 3: The value of hyperparametrs used in our experiments.

the neural network architecture in NAS. Similar to NAS where the goal is to optimize both network architecture and network weights, the goal of our problem is also to optimize both the robot and the policy. However, our problem is different from NAS in that our problem is much noisier than NAS, where the converged neural network performance in NAS usually has a noise of 1-2% while the policy reward rollouts in our problem can have a noise up to 90% depending on the reward design. Therefore, solutions that use Bayesian Optimization for NAS such as [14] will not be able to work in our problem. In fact, our proposed DEPS algorithm is closer to differentiable neural architecture search [13].

References

[1] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[2] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield, J. Kautz, and D. Fox. DexYCB: A benchmark for capturing hand grasping of objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[3] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020.

[4] A. Spurr, U. Iqbal, P. Molchanov, O. Hilliges, and J. Kautz. Weakly supervised 3d hand pose estimation via biomechanical constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 211–228. Springer, 2020.

[5] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[6] M. J. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[7] Y.-H. Wu, J. Wang, and X. Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022.

[8] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[10] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade. Towards generalization and simplicity in continuous control. *arXiv preprint arXiv:1703.02660*, 2017.

[11] X. Liu, D. Pathak, and K. M. Kitani. Revolver: Continuous evolutionary models for robot-to-robot policy transfer. *arXiv preprint arXiv:2202.05244*, 2022.

[12] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018.

[13] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[14] S. Cao, X. Wang, and K. M. Kitani. Learnable embedding space for efficient neural architecture compression. In *International Conference on Learning Representations*, 2018.